Viability of software AES in Wireless Sensor Networks

Ioan Deaconu, Andrei-Alexandru Musat
Automatic Control and Computers Faculty
University Politehnica of Bucharest,
{ioan.deaconu@cti.pub.ro}, {andrei.musat@cti.pub.ro}

***Abstract*—An experimental analysis on the viability of software AES in Wireless Sensor Networks**

Wireless sensor networks are a cheap and versatile solution for monitoring various environments. The data gathered by these networks is often used in research projects and scientifical experiments. It is necessary to ensure that a certain level of security and protection is applied on the transmitted data. Current software security methods implemented on wireless sensor networks do not necessarily take into consideration the lifetime and autonomy of the nodes. This paper presents an analysis of a software AES implementation on wireless sensor nodes and shows a comparison between the viability of such a method with regard to hardware AES.

Keywords: Wiereless Sensor Networks, Data Security, AES Encryption, Autonomy

I. Introduction

Wireless Sensors are low cost, low power devices optimized to perform custom tasks. They usually gather information from their surroundings and then send it to a base station server in order to be stored and processed. This communication is generally achieved using gateways. A gateway is usually connected to a more powerful device that can process the received information and take certain actions based on the results. The information transmitted by wireless sensors often represents sensitive data. For this reason, security protocols are implemented to prevent attacks that can intercept, replicate or alter the data.

Currently, security protocols in wireless sensor networks rely mostly on key based encryption algorithms. While this method can achieve great efficiency in terms of data security and protection, it also requires a certain level of computational power and is not always a task which is quickly executed. More so, in order to use such protocols, nodes must store all the necessary keys. Due to their design, wireless sensors often do not possess the necessary resources. They seldom have external memories attached to them and their processing power is limited to microprocessors which run at frequencies in the range of 1-20 MHz.

Another limitation of using this type of protocols to encrypt data is related to energy consumption. Usually, these sensors are powered by small batteries with a limited capacity. If the microprocessor has to perform intensive computations, these batteries will be drained in short amounts of time. Even equipping sensors with energy harvesting peripherals does not ensure that the battery lifespan is greatly increased.

The approach presented in this paper attempts to implement a more simple encryption algorithms which, combined with hardware encryption methods, can achieve an acceptable level of data security while ensuring that power consumption is kept to a minimum.

The proposed method relies on using available hardware AES ECB encryption in inter node communication and AES CBC for securing data.

This approach tries to find the balance point in the trade-off between security and energy consumption. While the data might not be protected as well as when key based algorithms are used, the energy consumption will be minimized thus increasing the life span of the sensor.

II. Related Work

Security is of prime importance in Wireless Sensor Networks. Nodes transfer important data between them and the cost of checking the validity and integrity of the transmitted packages is high, both from the energy consumption perspective as well as the necessary computational power perspective.

Possible attacks that might hinder the activity and integrity of a Wireless Sensor Network include:

- Wormhole attack: the attacker sends the received messages from one part of the network in a different part of the network. As a result, the nodes from both areas consider that the other nodes are neighbours and vice-versa.
- Blackhole/Sinkhole attack: the attackers makes itself more appealing from a routing point of view in order to receive all the messages from the network.
- Sybil attack: the attacker assumes the identity of one or more valid sensors[6].
- Selective forwarding attack: the attacker is able to intercept messages and drop certain packets or forward them [3].

- Hello Flood Attack: the attacker uses ”HELLO” packets to flood its neighbors in order to force the nodes to trust him.

No current security framework available for Wireless Sensor Networks offers complete protection against all types of attacks. However they offer protection against specific attacks. All of the following implementations rely on software encryption methods:

- SPINS - 2002 - The communication parties create independent keys for encryption and decryption and MAC keys for communication. It provides security against Data and Information Spoofing and Message Replay Attacks[7].
- LEAP - 2003 - The protocol implies that the nodes exchange more than one type of message between them.Thus, the framework uses 4 different keys. It provides security against ”HELLO” flood attacks, Sybil attacks and minimizes the consequences of spoofing, altering, replay routing information and selective forwarding attacks [9].
- TinySec - 2004 - The key is pre-deployed on the node, but it does not provide any solution for changing the key. If a node is compromised, the entire network will be compromised. It provides security against Data and Information Spoofing and Message Replay Attacks [4].
- LEAP+ - 2006 - It uses the same idea as LEAP, but the overhead is reduced. It provides security against Confidentiality and authentication, ”HELLO” flood attacks, Sybil attacks and minimizes the consequences of spoofing, altering, replay routing information and selective forwarding attacks [9].
- MiniSec - 2007 - Uses a counter IV mechanism. The counter is incremented locally and only the last bits of the counter are sent. It provides security against Authentication, Data Secrecy and Reply Attack [5].
- pDCS - 2009 - It uses 5 different keys to achieve data security. It provides security against Location and Query privacy [8].
- TinyKey - 2011 - An improvement of TinySec. It adds the key management system, in order to be able to change the key after the node is deployed. It provides security against Message authentication, confidentiality and integrity [1].
- ERP-DCS - 2013 - It proposes a different way of creating and storing keys when compared with pDCS. It provides security against Location and Query privacy [2].

III. Architecture

A. Hardware

The processing power and wireless capability of the wireless sensor nodes are provided by an Atmel ZigBit 900MHZ RF module. It contains an ATmega 1281V 8-bit microcontroller connected to an AT86RF212 RF Transceiver via a SPI interface. The Atmega 1281V is an low power 8 bit microcontroller that is connected to the onboard sensors of the node. In order to be able to transmit or secure the data, the microcontroller will communicate with the RF Transceiver. The Transceiver controller is a very low power chip, capable of sending data up to 6 km. Also, the Transceiver contains a security module compatible with AES-128. It supports hardware encryption and decryption for AES 128 ECB, but for the AES 128 CBC it is available only the hardware encryption.

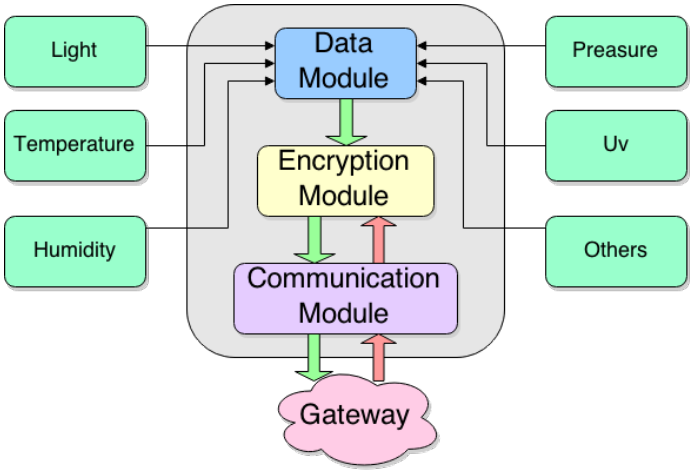

Figure 1: System Architecture

B. Software

From the software perspective, the architecture is composed of three modules:

- data module, that collects information from the sensors,
- encryption module,
- communication module.

Since the transceiver module is separated from the controller, from an efficiency perspective, it would be preferable to implement AES ECB and CBC algorithms as software services directly on the controller. Using this method, the transceiver shall be kept most of the time in an idle state, and the only component which will actively operate and drain the battery is the controller itself. While the controller is one of the components which a high power consumption rate, the implementation will contain optimizations meant to keep the number of encryption operations performed to a minimum. Moreover, it is important to filter data before packaging and sending it, in order to further reduce the number of encryption and decryption operations which the microcontroller of a node has to perform.

Overall, given the available hardware resources and architecture, the proposed implementation offers a software solution for ensuring data security and it also uses encryption methods which can easily be replaced with AES ones if ported on an hardware architecture in which the transceiver is incorporated in the microcontroller.

IV. Implementation

As stated in the Architecture chapter, the solution focuses on implementing the ECB and CBC algorithms in the Atmega controller firmware along with a series of optimizations meant to keep power consumption at a minimum.

A. ECB and CBC algorithms

Both the ECB and CBC algorithms represent block ciphers. These algorithms operate on fixed-length groups of bits called blocks. In order to encrypt/decrypt a block of data, a key is necessary. The difference between the two methods lies in how the key is applied.

For the ECB mode, each block is encrypted separately with the respective key, as show in the figure below. The disadvantage of this method is that identical plain text blocks shall generate identical cipher text blocks, which allows data patterns to emerge, thus making this encryption method relatively vulnerable to attacks.

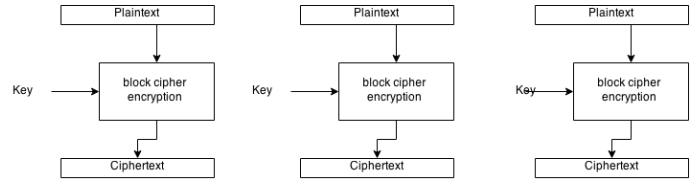

Figure 2: ECB mode of operation

In contrast, CBC mode encrypts a block by using information from either the previous block, or and initialization vector (IV). A xor operation is performed between the plain text and the previous cipher text or IV before applying the key. This method ensures pseudo-randomness and prevents patterns from emerging, thus making it more resilient to attacks.

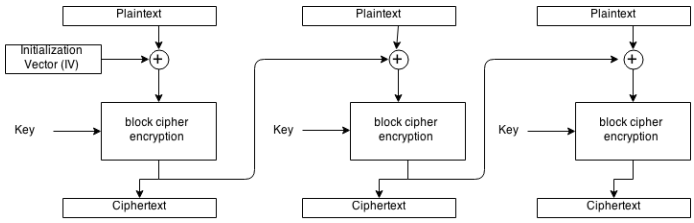

Figure 3: CBC mode of operation

B. Disadvantages of key based algorithms

There are two main reasons why the chosen encryption method is ECB/CBC. The first reason is that when ported to a different architecture which incorporates the transceiver on the same chip as the controller, the encryption and decryption can be safely handed over to the AES with minimal modifications and almost no degradation in power consumption.

The second reason is that an alternative solution would represent using key based encryption methods (both symmetric and asymmetric). Amongst the downsides of this particular method, we are mainly concerned with the following:

- Increased power consumption. Due to the nature and complexity of these algorithms as well as the number of operations which must be performed, the cost of such a solution is not optimal for low power sensors. By cost we refer to the necessary processing power and the total power consumption of the wireless sensor node.
- Memory limitations. The wireless sensor nodes were not designed to support external flash memory modules. The only available memory is the Atmega controller's flash memory. Since this memory is limited, writing additional data in it such as the node's private key and the public keys of other nodes would not be practical. A possible solution would be to add a special type of node which handles all data decryption and verification on the path towards the base station, but this would increase the size and complexity of the network.
- Key vulnerability. There are situations in which nodes will have to transmit public keys between themselves. Unless the respective keys are also encrypted prior to sending, should a man-in-the-middle type of attack occur, the keys can be intercepted and thus compromise the security of future transmissions.

C. Optimizations

Despite being a more efficient technique from the power consumption point of view, the software implementation can be improved even further by reducing the number of operations that must be performed on the microcontroller.

In the wireless sensor network, each node is identified by a MAC. In order to optimize the operations performed by a node when it receives a packet and also prevent attacks which try to introduce corrupt packets in the network, it shall first check if the source MAC is valid. This additional layer of security is cheap to implement and versatile.

First, each node shall contain a table of valid MACs stored in its local flash. The size of this table is proportional to the number of sensors in the network. Once a node has this table, upon receiving a packet, it shall drop it unless the source MAC matches an entry in the table and no decryption and verification operations shall be initiated.

Another software optimization is related to data processing before packets are assembled and the encryption operation can effectively begin. The data provided by the main sensor, the accelerometer, is often of no interest, as it presents no immediate changes in the frequency of an object's vibrations. Thus, in order to reduce the number of packets which a sensor node sends and prevent the flooding of the communication channel, this data is pre-processed.

Once the network is assembled and the nodes are powered on, they shall take a set of readings in order to calibrate

themselves to a certain level of vibrations. Then, whenever input is provided from the accelerometer, if that specific input is in range of the default values obtained during calibration, the information is dropped and no packet is formed. Furthermore, relevant data which is not in the range of the normal values is not sent after every reading. Instead, a series of N such readings from the accelerometer are processed into a median value, which is then transmitted into the network. Using this technique, the controller processes more useful information and the network will not flood itself.

V. Experimental Results

In this chapter we present the experimental setup which was used to test the transmition of encrypted data, as well as results related to the power efficiency of the sensors, the processing capabilities of the proposed solution and resilience towards flooding and DDoS attacks.

A. Experimental setup

The above mentioned tests were performed using a small network composed of 3 SparrowE wireless sensor nodes. In this setup, one of the SparrowE nodes was designated as the network coordinator while the other two nodes were plain sensors, as it can be seen in the figure below:

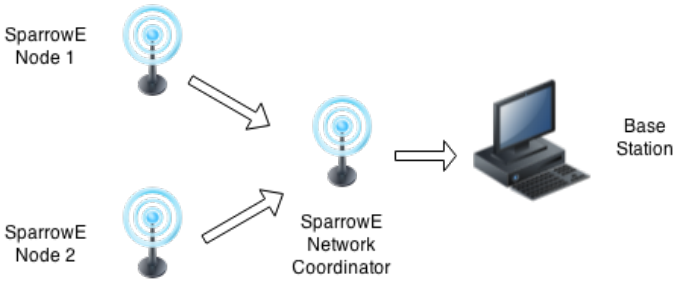

Figure 4: The SparrowE WSN experimental setup

The network coordinator is directly connected to the base station and its functions include analyzing incoming packets, decrypting the valid ones and passing the information on to the base station. The rest of the nodes are simply sensors which encapsulate and encrypt the data from their accelerometers and then broadcast it on the network.

B. Software ECB and CBC efficiency

The metric of interest is the number of encryption and decryption operations on a 16 bytes block which can be performed in a unit of time by the software implementation of the ECB and CBC algorithms and if it scales linearly like the AES implementation does.

In order to gather the required data, both EBC and CBC have been performed on data from the node's sensors for periods of time lasting 1, 2, 3, 4 and 5 seconds respectively. Then, the same was done using the AES implementations. The results can be observed in the tables shown below:

Table I: Number of operations performed by software/hardware ECB

Time	ECB Software Encryption	ECB Hardware Encryption	ECB Software Decryption	ECB Hardware Decryption
1	901	3552	520	3550
2	1805	7105	1044	7101
3	2707	10661	1562	10653
4	3604	14221	2084	14205
5	4508	17761	2603	17752

Table II: Number of operations performed by software/hardware CBC

Time	CBC Software Encryption	CBC Hardware Encryption	CBC Software Decryption	CBC Hardware Decryption
1	873	3550	509	3380
2	1747	7103	1023	6760
3	2623	10651	1530	10143
4	3495	14203	2038	13525
5	4367	17754	2546	16901

As it is seen in both tables, the results show that the software implementation also scales linearly, just like the hardware AES. Unfortunately, the software AES proves to be almost 7 times slower than the hardware AES. This is especially problematic in the case of data decryption, where the software AES will be much slower will take longer periods of time to detect possible corrupt packets. Furthermore, this will also cause the data from the sensors to be gathered less often and there is the danger that an interruption which occurs when sensor data is ready could suspend the decryption process.
One more issue encountered during the tests shows that due to the limited amount of RAM memory available on the controller, only 4Kb, a node cannot perform both software encryption and decryption at the same time.

C. Power consumption

Under normal circumstances, the Atmega controller on the SparrowE nodes functions at a supply voltage of 3.6V and a frequency of 8MHz. If the nodes run the encryption by using only the software method, then the transceiver shall mostly be turned off while the sensor is running and its power consumption is negligible.
While performing encryption operations, it has been measured that the controller has a medium power consumption of 36mW, which can reach peaks of 50.4mW, meaning it normally uses a current of 10mA which spikes at 14mA. If the hardware approach is used and the transceiver is turned on for large

periods of time, it adds a power consumption of 1.63mW-3.48mW, meaning an average current of 490uA.
Because software AES operations are performed slower than their hardware AES counterparts, it would seem that the total power consumption of the sensor nodes would be 7 times greater. However, when calculated together with the power consumption of all the other components, results show that using software AES decreases the sensor's autonomy by a factor of only 2 or 3.

D. Flooding and DDoS protection

Inside the wireless network running at 900MHz, the nodes communicate by broadcasting packets. The network coordinator node shall receive and analyze any packet which arrives on this channel. This opens up the risk of exposing the network to a DDoS attack. In order to determine how resilient the network is to such an attack and how much useful information the coordinator loses because it spends time analyzing and decrypting corrupt packets, such an attack has been simulated inside the experimental network.
The simulation is done by expanding the network to 5 nodes and having some of them transmit corrupt packets which are only meant to flood the network. The total size of these packets is 112 bytes. The size must be a multiple of 16 in order to be processed by the block encryption algorithms. Then, these packets will be sent at time intervals starting at the network's default 250ms and scaling down until the coordinator no longer receives any packets. The total number of packets sent during each iteration is 1500. The packet header is described below:

```
#define MAX_PACKET_DATA 17

typedef struct __attribute__((packed)){
        int16_t x;
        int16_t y;
        int16_t z;
} data_t;

typedef struct __attribute__((packed)){
        uint16_t node_id;
        uint32_t timestamp;
        uint16_t frame_index;
        uint16_t nr;
        data_t data[MAX_PACKET_DATA];
} frame_t;
```

The results can be seen in the figure below:
We can see that packet loss does not drastically increase in the interval of 250ms - 40ms. However, once we start sending packets once every 32ms or faster, the network becomes more and more flooded. When the transmission interval is below 27ms, the attacker's packets no longer reach the coordinator node (the attacker cannot send packets that fast and stops working), thus all the correct packets are processed and the loss is 0 percent. The overall packet loss for the 5 sensor network transmitting at intervals of 250 milliseconds is 2 percent.

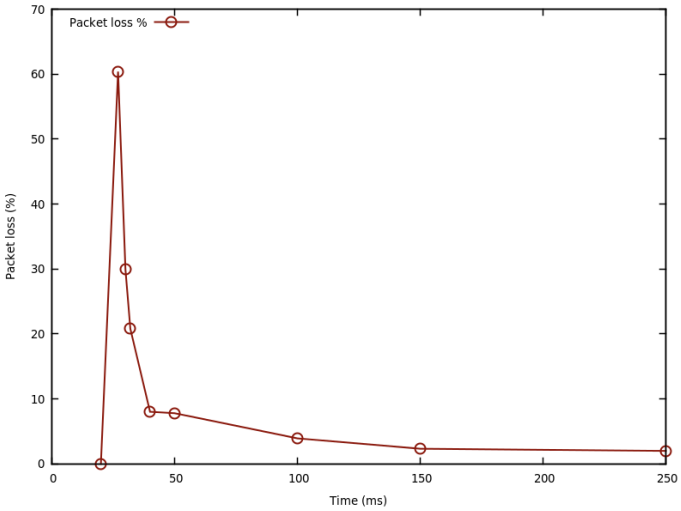

Figure 5: Graphic representation of packet loss correlated with interval between packet sending

VI. Conclusions and Future Work

A. Conclusion

In conclusion, over the course of the experiment we have observed that the software implementation of ECB and CBC encryption algorithms can not be used in low power wireless sensor networks as an alternative to the hardware AES because it is by its nature a slower solution and it does not integrate well with the concept of low power sensors.
We have seen that using this implementation, the number of encryption and decryption operations which can be performed in the unit of time is significantly smaller than when using hardware AES, up to a total of 7 times slower, especially in the case of the decryption operation.
A second drawback would be that the software implementation requires additional memory to store the tables used by the ECB and CBC methods to generate keys. This can take up to 21Kb of flash memory on the controller. In comparison, when using hardware AES, no such memory usage is required.
Going even further, the software implementation requires even more resources in order to perform both encryption and decryption operations on the same node. The 4Kb of RAM memory available on the node's controller is not sufficient to perform these tasks, one node being able to perform only decryption or only encryption.
The last test has shown that with the proposed headers for security, the coordinator (or gateway) node can still perform well even when flooded with a great number of corrupt packages meant to perform a DDoS attack and prevent the network from processing proper data.

B. Future Work

In addition to the AES implementation, there are a series of other encryption algorithms and methods which have been presented at the beginning of this paper. Future research and analysis can be performed to verify if any of those other

methods could perform better or on the same level as the hardware AES on the given experimental setup.

Furthermore, the overall energy efficiency of different implementations over the course of long periods of time can be studied. The goal would be to establish which encryption and security method is best suited for the SparroE wireless sensor nodes in order to ensure their operational lifetime is the maximum possible.

REFERENCES

[1] R. Doriguzzi Corin, G. Russello, and E. Salvadori. Tinykey: A lightweight architecture for wireless sensor networks securing real-world applications. In *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*, pages 68–75. IEEE, 2011.

[2] J.-M. Huang, S.-B. Yang, and C.-L. Dai. An efficient key management scheme for data-centric storage wireless sensor networks. *IERI Procedia*, 4:25–31, 2013.

[3] S. Kaplantzis, A. Shilton, N. Mani, and Y. A. Sekercioglu. Detecting selective forwarding attacks in wireless sensor networks using support vector machines. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 335–340. IEEE, 2007.

[4] C. Karlof, N. Sastry, and D. Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175. ACM, 2004.

[5] M. Luk, G. Mezzour, A. Perrig, and V. Gligor. Minisec: a secure sensor network communication architecture. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 479–488. ACM, 2007.

[6] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 259–268. ACM, 2004.

[7] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. E. Culler. Spins: Security protocols for sensor networks. *Wireless networks*, 8(5):521–534, 2002.

[8] M. Shao, S. Zhu, W. Zhang, G. Cao, and Y. Yang. pdcs: Security and privacy support for data-centric sensor networks. *Mobile Computing, IEEE Transactions on*, 8(8):1023–1038, 2009.

[9] S. Zhu, S. Setia, and S. Jajodia. Leap+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(4):500–528, 2006.